Apple IIGS
#76: Miscellaneous Resource Formats

Revised by: Matt Deatherage May 1992
Written by: Matt Deatherage, C.K. Haun, Llew Roberts & Dave Lyons January 1990

This Technical Note describes resource structure formats for previously-unpublished types.

Changes since December 1991: Added information on `rFont` resources. Clarified the note about `rVersion` resources to note that version numbers must increase with subsequent releases for the Finder's benefit.

The format used to describe the resources is similar to that used in File Type Notes, where the offsets, given in the form (+xxx), determine the offset from the beginning of the resource.

Sampled Sound Resource (Type: $8024, rSoundSample)

The following describes the Sampled Sound resource format. It consists of a ten-byte header followed by the sample data bytes.

Format	(+000)	**Word**	This must always be zero.
Wave Size	(+002)	**Word**	Sample size in pages (256 bytes per page). For example, an 8K sample takes 32 pages; a 128K sample requires $200 pages.
Rel Pitch	(+004)	**Word**	The high byte of this word is a semitone value; the low byte is a fractional semitone. These values are used to tune the sample to correct pitch. See HyperCard IIGS Technical Note #3, Pitching Sampled Sound.
Stereo	(+006)	**Word**	The output channel for this sound is in the low nibble of this word.
Sample rate	(+008)	**Word**	The sampling rate of the sound, in Hertz (Hz).
Sound	(+010)	**Bytes**	The sampled sound data. The bytes are all 8-bit samples. The sample starts here and continues until the end of the resource.

The resource compiler template follows:

```
#define rSoundSample  $8024

/*---------------------- rSoundSample --------------------*/
type rSoundSample {
       integer;              /* format */
       integer;              /* wave size */
       hex integer;          /* rel pitch */
       integer;              /* stereo channel */
       unsigned integer;     /* sample rate */
       hex string;           /* raw 8 bit sound data */
};
```

Cursor Resource (Type: $8027, rCursor)

The following describes the Cursor resource format:

height	(+000)	**Word**	The height of the cursor, in pixels.
width	(+002)	**Word**	The width of the cursor, in **Words**.
image	(+004)	**Bytes**	The image of the cursor. There are height*width **Words** in the cursor, or twice that many **Bytes**.

We will call the first byte beyond the image offset "ei" for "end of image."

mask	(+ei)	**Bytes**	The mask of the cursor. This is the same size as the image.

We will call the first byte beyond the mask offset "em" for "end of mask."

hotSpotY	(+em)	**Word**	The cursor's Y "hot spot."
hotSpotX	(+em+2)	**Word**	The cursor's X "hot spot."
flags	(+em+4)	**Flag Word**	Cursor flags: Bit 7: 1 = 640 Mode, 0 = 320 Mode All other bits are reserved and must be zero.
reserved	(+em+6)	**8 Bytes**	Reserved, must be zero.

The resource compiler template follows:

```
#define rCursor       $8027

/*---------------------- rCursor --------------------*/
type rCursor {
    height :
        hex integer;            /* height */
    width :
        hex integer;            /* width in words */
        hex string[2*$$Word(height)*$$Word(width)];    /* cursor image */
        hex string[2*$$Word(height)*$$Word(width)];    /* cursor mask */
        hex integer;            /* hotspot Y */
        hex integer;            /* hotspot X */
        hex integer;            /* flags */
        hex longint = 0;        /* reserved */
        hex longint = 0;        /* reserved */
    };
};
```

Following is a simple cursor example:

```
resource rCursor(1,fixed) {
        5,     /* height */
        2,     /* width */
        $"ffff0000"
        $"f00f0000"
        $"f00f0000"
        $"f00f0000"
        $"ffff0000",

        $"ffff0000"
        $"ffff0000"
        $"ffff0000"
        $"ffff0000"
        $"ffff0000",

        2,     /* hotspot Y */
        2,     /* hotspot X */
        $80    /* 640 mode */
};
```

Note that the resource is marked `fixed` so that its handle can be dereferenced and passed to `SetCursor`.

Version Resource (Type: $8029, rVersion)

Files may include a version resource with ID=1 for display by programs such as the Finder. All `rVersion` resource IDs other than 1 are **reserved** for future definition. The following describes the version resource format:

version	(+000)	**Long**	The application's version number, in Apple IIGS Long Version format. See Apple IIGS Technical Note #100, VersionVille, for more details.
country	(+004)	**Word**	An international country version code. Possible values are as follows:

verUS	0
verFrance	1
verBritain	2
verGermany	3
verItaly	4
verNetherlands	5
verBelgiumLux	6
verSweden	7
verSpain	8
verDenmark	9
verPortugal	10
verFrCanada	11
verNorway	12
verIsrael	13
verJapan	14
verAustralia	15
verArabia	16
verFinland	17
verFrSwiss	18
verGrSwiss	19
verGreece	20
verIceland	21

			verMalta 22 verCyprus 23 verTurkey 24 verYugoslavia 25 verIreland 50 verKorea 51 verChina 52 verTaiwan 53 verThailand 54
name	(+006)	**String**	Pascal string containing the desired name. May be the null string.
moreInfo	(+xxx)	**String**	Additional information to be displayed, such as a copyright notice. May be the null string. Recommended maximum length is about two lines of 35 characters each. May contain a carriage return (character $0D).

The resource compiler template follows:

```
#define rVersion      $8029

// Equates for the country code of an rVersion resource

#define Region                                          \
       verUS, verFrance, verBritain, verGermany,        \
       verItaly, verNetherlands, verBelgiumLux,         \
       verFrBelgiumLux = 6, verSweden, verSpain,        \
       verDenmark, verPortugal, verFrCanada, verNorway, \
       verIsrael, verJapan, verAustralia, verArabia,    \
       verArabic=16, verFinland, verFrSwiss, verGrSwiss,\
       verGreece, verIceland, verMalta, verCyprus,      \
       verTurkey, verYugoslavia, verYugoCroatian = 25,  \
       verIndia = 33, verIndiaHindi = 33, verPakistan,  \
       verLithuania = 41, verPoland, verHungary,        \
       verEstonia, verLatvia, verLapland, verFaeroeIsl, \
       verIran, verRussia, verIreland = 50, verKorea,   \
       verChina, verTaiwan, verThailand                 \

/*------------------- rVersion ------------------*/
type rVersion {
   ReverseBytes {
       hex byte;                                    // Major revision in BCD
       hex bitstring[4];                            // Minor vevision in BCD
       hex bitstring[4];                            // Bug version
       hex byte   development = 0x20,               // Release stage
                  alpha = 0x40,
                  beta = 0x60,
                  final = 0x80, /* or */ release = 0xA0;
       hex byte;                                    // Non-final release #
   };
   integer Region;                          // Region code
   pstring;                                         // Short version number
   pstring;                                         // Long version number
};
```

Following is a simple version example for “Super Graphics Destroyer”, version 2.0:

```
resource rVersion(1) {
       { $02,$0,$0,release,$00 },          /* version 2.0 release */
       verUS,                              /* US version */
       "Super Graphics Destroyer",         /* our app's name */
       "© 1991 Pretty as a Picture, Inc."  /* the copyright notice */
};
```

Note: For compatibility with the Finder, keep the `name` field identical across different versions of the same application, and make sure the `version` field increases on each later version released to your customers. The `moreInfo` field is not critical; if it changes between versions, it's no big deal.

Comment Resource (Type: $802A, rComment)

Files may include a comment resource with ID=1 for display by programs such as the Finder. All `rComment` resource IDs other than 1 are **reserved** for future definition. The following describes the comment resource format:

text	(+000)	**Bytes**	The comment. This is unformatted, 8-bit text suitable for displaying by a desktop program. No length limit is imposed by this resource format, although a practical limit of a few hundred characters is recommended.

The resource compiler template follows:

```
#define rComment      $802A

/*---------------------- rComment --------------------*/
type rComment {
    string;
};
```

Tagged Strings Resource (Type: $802E, rTaggedStrings)

A tagged strings resource lists pairs of Word values and Pascal strings.

count	(+000)	**Word**	Number of word/string pairs in this resource.
firstWord	(+002)	**Word**	Word value of first pair.
firstString	(+004)	**String**	Pascal string of first pair.
secondWord	(+xxx)	**Word**	Word value of second pair.
secondString	(+yyy)	**String**	Pascal string of second pair.
…			

The resource compiler template follows:

```
#define rTaggedStrings              $802E

/*------------------- rTaggedStrings --------------------*/

type rTaggedStrings {
              integer = $$Countof(StringArray);
              array StringArray {
                            hex integer;          /* Key integer */
                            pstring;                    /* String */
              };
};
```

Following is a simple `rTaggedStrings` example:

```
resource rTaggedStrings(1) {{
       $0050, "red",
       $0033, "green",
       $0100, "blue"
}};
```

Pattern List Resource (Type: $802F, rPatternList)

A pattern list resource contains zero of more 32-byte QuickDraw II patterns. (This resource type exists for your convenience. The System Software contains no direct support for resources of this type.)

firstPattern	(+000)	**32 Bytes**	First QuickDraw II pattern structure.
secondPattern	(+032)	**32 Bytes**	Second QuickDraw II pattern structure.
…			

The resource compiler template follows:

```
#define rPatternList         $802F

/*-------------------- rPatternList ---------------------*/

type rPatternList {
       array {
              array[32] {
                     hex byte;
              };
       };
};
```

Rectangle List Resource (Type: $C001, rRectList)

The rectangle list (type `rRectList`) is provided to allow an extensible, easily modifiable collection of QuickDraw II rectangle structures. This capability can enhance a developer's ability to modify on-screen displays without recompiling an entire application. This resource also enables easier cross-development and parallel development for the Apple IIGS and the Macintosh.

The following describes the rectangle list resource format:

count	(+000)	**Word**	Number of rectangles in this resource.
firstRectangle	(+002)	**8 Bytes**	First QuickDraw II rectangle structure.
…			Rectangles, eight bytes each.
lastRectangle	(+002+(8*(count-1)))	**8 Bytes**	Last QuickDraw II rectangle structure.

The resource compiler template follows:

```
#define rRectList    $C001
type rRectList {
        integer = $$Countof(RectArray);
        array RectArray {
            Rect;
            };
};
```

Print Record Resource (Type: $C002, rPrintRecord)

As a convenience for applications, a print record may be included as a resource of type $C002 (`rPrintRecord`). If more than one of these resources is present, the one to use as the document's primary print record is the first one. You can get this resource's ID by calling `GetIndResource` with type `rPrintRecord` and index 1. Storing the primary print record with ID = 1 is a good way to start.

Since the print record is filled in and interpreted by the printer driver, you can't always programmatically set options that are driver-specific. For example, although the ImageWriter driver stores the color-vs.-black and white option in one place, not all color printers will do the same thing. If you want to use driver-specific options on many printers, you can use those printer drivers to create print record that reflect the options you want and store those records as resources. Then, if you need a pre-initialized print record with the options you want, you may already have one.

print record	(+000)	**160 Bytes**	The print record. The handle to this resource is suitable for passing to any Print Manager call that requires a print record handle.

Since applications shouldn't create print records from scratch, but rather allow printer drivers to fill them in with `PrDefault` and `PrVerify`, the resource compiler template that follows only allocates 160 bytes for storage.

```
#define rPrintRecord  $C002

/*-------------------- rPrintRecord ------------------*/
type rPrintRecord {
       array[160] {
              hex byte;
       };
};
```

Font Resource (Type: $C003, rFont)

Some applications wish to keep fonts with the application itself instead of in a separate font file. This isn't always advisable—fonts not in font files can't be easily used in other applications, which may confuse users who, for example, use text-editing desk accessories and can't get at certain fonts except in certain applications. Also, fonts not in the Fonts directory must be completely memory-resident where normal Fonts are only loaded from disk when they're needed.

Nevertheless, in some cases keeping fonts inside an application file is necessary or desirable. In such cases, the `rFont` resource is a convenient way to keep a font around. The resource is a QuickDraw II Font, as defined in *Apple IIGS Toolbox Reference*, Volume 2. The font family name is the resource's name—the same Pascal string that normally precedes the QuickDraw II font record in the font file.

font	(+000)	**Bytes**	The font record, without the font family name.

Since the font record is a bunch of variable-sized tables, and since you probably want to use a font editor (and not the resource compiler) to create fonts, the following resource compiler template isn't very revealing.

```
/*-------------------- rFont ------------------*/

type rFont {
       hex string;
};
```

Further Reference

- *Apple IIGS Toolbox Reference*, Volumes 1–3
- Apple IIGS Technical Note #100, VersionVille
- HyperCard IIGS Technical Note #3, Pitching Sampled Sounds